A Bayesian Analysis of Unobserved Component Models using Ox

Bos, C.S.

published in
Journal of Statistical Software
2011

DOI (link to publisher)
10.18637/jss.v041.i13

document version
Publisher's PDF, also known as Version of record

Journal of Statistical Software

May 2011, Volume 41, Issue 13. http://www.jstatsoft.org/

A Bayesian Analysis of Unobserved Component Models Using Ox

Charles S. Bos
VU University Amsterdam

Abstract

This article details a Bayesian analysis of the Nile river flow data, using a similar state space model as other articles in this volume. For this data set, Metropolis-Hastings and Gibbs sampling algorithms are implemented in the programming language Ox. These Markov chain Monte Carlo methods only provide output conditioned upon the full data set. For filtered output, conditioning only on past observations, the particle filter is introduced. The sampling methods are flexible, and this advantage is used to extend the model to incorporate a stochastic volatility process. The volatility changes both in the Nile data and also in daily S&P 500 return data are investigated. The posterior density of parameters and states is found to provide information on which elements of the model are easily identifiable, and which elements are estimated with less precision.

Keywords: state space methods, unobserved components, Bayes, stochastic volatility.

1. Introduction

In this article we analyze the Nile data from a Bayesian perspective using Ox (Doornik 2009). The initial model in Section 2 is the standard local level model. As the model is linear and Gaussian, the Kalman filter equations (see Commandeur, Koopman, and Ooms 2011) allow directly for a classical analysis. The added value of a Bayesian analysis, however, is that full information on the uncertainty of the parameters is obtained from the posterior density, and that problems with multimodality of the (likelihood or posterior) density surface are tackled more easily.

The aim of this article is not to provide the most thorough analysis of the Nile river flow data set, but instead describe the basic steps of a Bayesian analysis of the model, together with details on the implementation of the algorithms in Ox. The algorithms are chosen for the insight they give, not necessarily always for being the most advanced, quickest, or most

precise option at hand. The article is complemented with a set of programs (see the listing in Appendix B) for performing the analysis.

Two different sampling algorithms are presented. The Metropolis-Hastings algorithm (Section 2.2) is appropriate for the basic local level model. For illustrative purposes, a Gibbs sampling scheme is proposed in Section 2.3. Section 2.4 presents a particle filter for the model in which filtered estimates of the level component are evaluated. A filtered estimate is conditional on the past and concurrent observations while the smoothed estimate is based on all observations. The Bayesian sampling schemes from Sections 2.2 and 2.3 produce smoothed estimates. Using the estimate of the likelihood from the particle filter, the marginal likelihood is also obtained. The Gibbs sampler and particle filter are used in Section 3 to extend the local level model allowing for stochastically varying variance in the observation equation.

The algorithms are applied first on the Nile data set, with empirical results in Section 2.5. Further empirical results for the case with stochastic volatility are provided in Section 3.3, and an application on a financial data set concerning daily S&P 500 returns follows in Section 4. Section 5 provides concluding remarks.

2. The local level model applied to the Nile data

2.1. Setting up the model and prior

In order to analyse the local level model for the Nile data using a Markov chain Monte Carlo (MCMC) setup, first an analysis of the parameters occuring in the model and their possible sampling schemes is needed. For ease of reference, the model that interests us here is

$$y_t = \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim \text{NID}(0, \sigma_\varepsilon^2), \tag{1}$$

$$\mu_{t+1} = \mu_t + \xi_t, \qquad \xi_t \sim \text{NID}(0, \sigma_\xi^2), \tag{2}$$

for $t = 1, \ldots, T$. The model contains two parameters, $\theta = (\sigma_\varepsilon, \sigma_\xi)$, and one vector of unobserved states $\mu = (\mu_1, \ldots, \mu_T)$.

A Bayesian analysis is performed by combining prior information on the parameters with the information from the likelihood function, according to Bayes' formula

$$\mathsf{P}(\theta|Y_T) \propto \mathcal{L}(Y_T;\theta)\pi(\theta), \tag{3}$$

where $\mathcal{L}(Y_T;\theta)$ is the likelihood of data $Y_T = (y_1, \ldots, y_T)$ evaluated at the vector of parameters θ, and $\pi(\theta)$ is the prior density.

As we shall see later, a convenient prior density for the standard deviations σ_ε and σ_ξ is the Inverted Gamma-1 (IG-1) density (see e.g., Bauwens, Lubrano, and Richard 1999),

$$f_{\text{IG-1}}(\sigma; r, a) = \frac{2a^r}{\Gamma(r)\sigma^{2r+1}} \exp\left(-\frac{a}{\sigma^2}\right)$$

with expectation and variance

$$\mathsf{E}(\sigma) = \sqrt{a}\frac{\Gamma\left(r - \frac{1}{2}\right)}{\Gamma(r)}, \quad (r > \frac{1}{2}), \qquad \mathsf{VAR}(\sigma) = \frac{a}{(r-1)} - \mathsf{E}(\sigma)^2, \quad (r > 1).$$

This IG-1 density is the so-called *conjugate* density, implying that the posterior density of the parameters in this model is available in closed form if this specific form of prior density is chosen.

For the present linear Gaussian model, the likelihood function $\mathcal{L}(Y_T;\theta)$ can be calculated from the output of the Kalman filter (KF, see Harvey 1989; Durbin and Koopman 2001). Therefore, a sample from the posterior density of $\sigma_\varepsilon, \sigma_\xi | Y_T$ can be obtained using a Metropolis-Hastings (MH) approach (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953; Chib and Greenberg 1995). This will be detailed in Section 2.2. Alternatively, a Gibbs sampling approach (Smith and Gelfand 1992) with data augmentation would consider the state μ as part of the parameter space, and successively draw from the full-conditional densities $\mathsf{P}(\mu|Y_T, \sigma_\varepsilon, \sigma_\xi)$, $\mathsf{P}(\sigma_\varepsilon|Y_T, \mu, \sigma_\xi)$ and $\mathsf{P}(\sigma_\xi|Y_T, \mu, \sigma_\varepsilon)$. Such an approach has the advantage that also a sample from the posterior density of the states is obtained, and that it is easily extended to include elements breaking the linearity or Gaussianity of the model. This latter approach is presented in Section 2.3.

Both the MH and Gibbs samplers result in a posterior density of the parameters conditional on the full data set Y_T. At times, the filtered states $\mu_t|Y_t, \sigma_\varepsilon, \sigma_\xi$ are of interest. Section 2.4 presents a particle filter for this purpose, and is followed by Section 2.5 with empirical results on the Nile data.

2.2. Implementing a random walk Metropolis approach

For the Metropolis-Hastings approach, the general algorithm follows the steps:

1. Initialize, start with a vector of parameters $\theta^{(0)}$, set $i = 1$;
2. Draw a candidate value $\theta^* \sim q(\theta|\theta^{(i-1)})$;
3. Accept the new draw with probability

$$\alpha_{\mathrm{MH}}(\theta^*, \theta^{(i-1)}) = \min\left[\frac{\mathsf{P}(\theta^*|Y_T)\, q(\theta^{(i-1)}|\theta^*)}{\mathsf{P}(\theta^{(i-1)}|Y_T)\, q(\theta^*|\theta^{(i-1)})}, 1\right]. \tag{4}$$

   If the draw is accepted, set $\theta^{(i)} = \theta^*$, else $\theta^{(i)} = \theta^{(i-1)}$;
4. Increase i, and repeat from Step 2. until the sample is sufficiently large.

The candidate density $q(\theta|\theta^{(i-1)})$ can be chosen freely, though a density which is related to the target density would lead to better acceptance rates. Here, a random walk Metropolis scheme is used, taking the candidate $q(\theta|\theta^{(i-1)})$ to be a multivariate normal density with expectation $\theta^{(i-1)}$ and a prespecified covariance matrix,

$$q(\theta|\theta^{(i-1)}) \sim \mathrm{NID}\left(\theta^{(i-1)}, \Sigma_q\right).$$

As this candidate density is symmetric, it drops from Equation 4, simplifying the sampling further.

Basic code for the sampler itself could be implemented as in Tables 1–2, using the programming language Ox (Doornik 2009) and the package **SsfPack** (Koopman, Shephard, and Doornik 1999, 2008). The routine `EstBayesMH()` of Table 1 draws a sample of size `iS`, using random

```
EstBayesMH(const amTheta, const avP, const vY, const mPrior,
           const iS, const vSRW)
{
  decl i, iA, vPC, dLnDens, dLnDensC, dU;

  iA= 0;                              // Initialize acceptance counter
  amTheta[0]= zeros(2, iS);                // Storage space for sample
  dLnDens= LnPosterior(avP[0], vY, mPrior);      // Prepare posterior
  for (i= 0; i < iS; ++i)
    {
      vPC= avP[0] + rann(2, 1) .* vSRW;             // Draw candidate
      dLnDensC= LnPosterior(vPC, vY, mPrior);

      dU= ranu(1, 1);                               // Check acceptance
      if (dU < exp(dLnDensC - dLnDens))
        {
          ++iA;                                        // Accept draw
          avP[0]= vPC;
          dLnDens= dLnDensC;
        }
      amTheta[0][][i]= avP[0];                          // Store draw
    }
  return iA/iS;                                 // Return acceptance rate
}
```

Table 1: The random walk Metropolis scheme in Ox.

walk candidate standard deviations as specified in `vSRW`, and using a function `LnPosterior()` providing an evaluation of the logarithm of the posterior density (see Table 2). As output, the `EstBayesMH()` routine places the posterior sample of θ into a matrix at the address `amTheta`.[1] It returns the fraction of accepted drawings. Before discussing the results, the next section first provides a simple implementation of a Gibbs approach using data augmentation, followed by a particle filter scheme.

2.3. Implementing a Gibbs approach with data augmentation

For the second approach, the parameter space is augmented with the vector of states $\mu = (\mu_1, \ldots, \mu_T)$, and successively a draw is made from the full conditional densities. The algorithm follows the steps:

1. Initialize, start with vectors $\theta^{(0)}$ and $\mu^{(0)}$, and set $i = 1$;

[1] In Ox, a routine can receive the address `amTheta= &mTheta` as a function argument. At the location of the address, `amTheta[0]`, the routine can leave output values. The routine `EstBayesMH()` for instance fills the matrix column-by-column referring to `amTheta[0][][i]`, for successive value of `i`.

```
LnPosterior(const vP, const vY, const mPrior)
{
  decl ir, i, dLnPost, dVar, mPhi, mOmega, mSigma;

  mPhi= <1; 1>;                      // Prepare local level model
  mOmega= diag(sqr(vP));
  mSigma= <-1; 0>;

  ir= SsfLik(&dLnPost, &dVar,          // Likelihood evaluation
             vY, mPhi, mOmega, mSigma);
  for (i= 0; i < 2; ++i)                      // Prior evaluation
    dLnPost+= lndensigamma1(vP[i], mPrior[i][0], mPrior[i][1]);

  return ir ? dLnPost : M_NAN;
}
```

Table 2: Evaluating the log posterior density.

2. Sample new states $\mu^{(i)} \sim \mathsf{P}(\mu|Y_T, \theta^{(i-1)})$ using the simulation smoother. For a description see Appendix A or Durbin and Koopman (2002);

3. Sample $\sigma_\varepsilon^{(i)} \sim \mathsf{P}(\sigma_\varepsilon|Y_T, \mu^{(i)}, [\sigma_\xi^{(i-1)}])$. This density is a IG-1 density, as can be seen from

$$\begin{aligned}
\mathsf{P}(\sigma_\varepsilon|Y_T, \mu^{(i)}, [\sigma_\xi^{(i-1)}]) &\propto \mathcal{L}(Y_T; \mu^{(i)}, \sigma_\varepsilon)\pi(\sigma_\varepsilon)\pi(\sigma_\xi^{(i-1)}) \\
&\propto \sigma_\varepsilon^{-T} \exp\left(-\frac{1}{2\sigma_\varepsilon^2}\sum_{i=1}^{T}\left(y_t - \mu_t^{(i)}\right)^2\right) \frac{2a_\varepsilon^{r_\varepsilon}}{\Gamma(r_\varepsilon)\sigma_\varepsilon^{2r_\varepsilon+1}} \exp\left(-\frac{a_\varepsilon}{\sigma_\varepsilon^2}\right) \\
&\propto \frac{1}{\sigma_\varepsilon^{2r_\varepsilon+T+1}} \exp\left(-\frac{1}{\sigma_\varepsilon^2}\left(a_\varepsilon + \frac{1}{2}\sum_{i=1}^{T}\left(y_t - \mu_t^{(i)}\right)^2\right)\right) \\
&\equiv \text{IG-1}\left(r = r_\varepsilon + \frac{T}{2}, a = a_\varepsilon + \frac{1}{2}\sum\left(y_t - \mu_t^{(i)}\right)^2\right);
\end{aligned}$$

4. Sample a new $\sigma_\xi^{(i)} \sim \mathsf{P}(\sigma_\xi|\mu^{(i)}, [Y_T, \sigma_\varepsilon^{(i)}])$. This density is again a IG-1 density, with

$$\mathsf{P}(\sigma_\xi|\mu^{(i)}, [Y_T, \sigma_\varepsilon^{(i)}]) = \text{IG-1}\left(r = r_\xi + \frac{T-1}{2}, a = a_\xi + \frac{1}{2}\sum_{i=2}^{T}\left(\mu_t^{(i)} - \mu_{t-1}^{(i)}\right)^2\right);$$

5. Increase i, and repeat from Step 2. until the sample is sufficiently large.

The implementation of the algorithm is nothing more than a loop, similar to Table 1, around the three samplers introduced above. An implementation of the simulation smoother code is presented in Appendix A, with detailed explanations. For the standard deviations, a routine as in Table 3 can draw new values from the IG-1 distribution. Notice that the Inverted

```
SimSigmaIG1(const vE, const vRA)
{
  decl dr, da, dH;
  dr= vRA[0] + sizerc(vE)/2;
  da= vRA[1] + sumsqrr(vE)/2;

  dH= rangamma(1, 1, dr, da);   // Simulate precision 1/s2 ~ G(r, a)
  return 1 / sqrt(dH);      // Return standard deviation ~ IG1(r, a)
}
```

Table 3: Simulating the standard deviations in Ox.

Gamma-1 density relates directly to the more standard Gamma density: If $h \sim \text{Gamma}(r, a)$, then $\sigma = 1/\sqrt{h} \sim \text{IG-1}(r, a)$. This relationship is applied here to enable the use of the Ox function `rangamma()` for generating the standard deviation.

The Gibbs algorithm results in a sample from the state vector μ and parameters $\sigma_\varepsilon, \sigma_\xi$, all of them conditional on the full data set $Y_T = (y_1, \ldots, y_T)$. For the MH approach of the previous section, no posterior sample of the states was available. What however is missing for both the MH and Gibbs samplers, is information on the filtered quantities, like the prediction errors or the filtered state, and also the value of the likelihood. In the case of the Nile data, with the basic local level model, these quantities can be obtained from the KF. The next section describes the basic approach of particle filtering, which can deliver similar output on filtered states and the likelihood as the KF, but is not restricted to linear and Gaussian state space models.

2.4. A particle filter

For the fully linear unobserved components model (1)–(2), the standard Kalman filter routines can be used to extract filtered quantities. If the model is extended to include non-linear or non-Gaussian components as well, then a precise Kalman filter is no longer available, and an approximative simulation method has to be used. Cappé, Godsill, and Moulines (2007) give an overview of possible particle filter implementations. A simple version is provided here.

The main idea of the particle filter is to sample a cloud of particles, $m_t^{(i)}, i = 1, \ldots, M$ such that they together describe the density of the state μ at time t given the data up to and including y_t. The most basic particle filter is described by the following steps:

1. To initialize, simulate $m_1^{(i)}|y_1 \sim \text{NID}(y_1, \sigma_\varepsilon^2), i = 1, \ldots, M$, where it is assumed that the prior density of the states is diffuse, such that the first observation contains all possible information on the first state. Set weights $\omega_1^{(i)} \equiv 1/M$, as the particles are sampled precisely from the correct density. Set $t = 2$, and commence the loop;

2. Choose M particles with replacement from the current set, using weights $\omega_{t-1}^{(i)}$, and call these $\tilde{m}_{t-1}^{(j)}, j = 1, \ldots, M$;

3. Propagate the particles according to a (candidate) transition density, sampling

$$m_t^{(i)} \sim q(m_t|\tilde{m}_{t-1}^{(i)}, y_t),$$

with corresponding weights

$$\tilde{\omega}_t^{(i)} = \frac{p(y_t|m_t^{(i)})p(m_t^{(i)}|m_{t-1}^{(i)})}{Mq(m_t^{(i)}|m_{t-1}^{(i)}, y_t)}, \qquad \omega_t^{(i)} = \frac{\tilde{\omega}_t^{(i)}}{\sum_{j=1}^{M}\tilde{\omega}_t^{(j)}}, \qquad i = 1, \ldots, M;$$

4. Increase t, and repeat from Step 2. while $t \leq T$.

The candidate transition density $q(m_t|\tilde{m}_{t-1}^{(i)}, y_t)$ can make use of the observation y_t, such that a set of new particles is sampled which take the newest observation into account. A simple choice, however, is to take

$$q(m_t|\tilde{m}_{t-1}^{(i)}, y_t) \equiv p(m_t|\tilde{m}_{t-1}^{(i)}) \sim \text{NID}\left(m_{t-1}^{(i)}, \sigma_\xi^2\right).$$

In the latter case, the weights in Step 3. simplify, and only contain the likelihood contributions. The alternative is to implement a single step of the filtering equation of the Kalman filter, and sample from the resulting density. This would lead to a better set of particles, with more uniformly distributed weights, at the cost of a slight increase in difficulty of implementation. In Bos and Shephard (2006) the particles even consist of the first two moments of the states, updated using the Kalman equations, and hence an exact filter results for a linear and Gaussian unobserved components model.

In case a certain observation is not available, the weights can obviously not be calculated. In such a situation, equal weights are set, and the algorithm continues. This is the way that a forecast from the state can be made, straight from the particle filter.

From the particle filter, output on the filtered particles can be collected. A basic implementation of the particle filter is provided in Table 4. The mean and standard deviation of the particles is collected during the filter, and based on the propagated particles before resampling using weights ω_t (e.g., $\overline{m}_t = \sum_i \omega_t^{(i)} m_t^{(i)}$). Alternatively, the mean and standard deviation could be constructed from the particles after resampling (i.e., $\overline{m}_t = \frac{1}{M}\sum_i \tilde{m}_t^{(i)}$), but this latter option induces a loss of efficiency due to the resampling.

One of the outputs of the above particle filter is the loglikelihood contribution of each observation, which equals the logarithm of the sum of non-normalized weights (see Pitt and Shephard 1999). The sum of the loglikelihood contributions delivers the loglikelihood of the data for a given set of parameters, integrating out the unobserved states. The likelihood $\mathcal{L}(Y_T;\theta)$ in turn can be used as input for computing the marginal likelihood $M(Y_T|\,\mathcal{M})$ of a model $\mathcal{M}$, which is the likelihood of the model integrating out the parameters with respect to the prior $\pi(\theta)$:

$$M(Y_T|\,\mathcal{M}) = \int_\theta \mathcal{L}(Y_T;\theta)\pi(\theta)\partial\theta \equiv \frac{\mathcal{L}(Y_T;\theta)\pi(\theta)}{\mathsf{P}(\theta|Y_T)}. \tag{5}$$

In this formula, the value of the posterior density $\mathsf{P}(\theta|Y_T)$ needs to be computed including the integrating constant. Several methods exist, but a simple one uses a Laplace approximation

```
PartFiltLL(const amMS, const avLL, const avVarW,
           const vP, const vY, const iM)
{
  ...
  vM= vY[0] + dSEps * rann(1, iM);            // Init m ~ N(y, s2Eps)
  vW= ones(1, iM)/iM;

  avLL[0][0]= 0;                // No likelihood contribution of y(0)
  amMS[0][][0]= meanr(vM)|sqrt(varr(vM));      // Store mean and sdev
  for (i= 1; i < iT; ++i)
    {
      vSel= ranmultinomialindex(iM, vW);                  // Resample
      vM= vM[vSel] + dSXi * rann(1, iM);                 // Propagate

      vE= (vY[i] - vM)/dSEps;                       // Compute weights
      vW= exp(-0.5*sqr(vE)) / (iM * dSEps * sqrt(M_2PI));

      avLL[0][i]= log(sumr(vW)); // Log of sum non-normalized weights
      vW/= sumr(vW);                                // Normalize weights

      if (ismissing(vY[i]))             // Reset weights if Y missing
        vW= ones(1, iM)/iM;
      amMS[0][0][i]= sumr(vW .* vM);          // Store mean and sdev
      amMS[0][1][i]= sqrt(sumr(vW .* sqr(vM)) - sqr(amMS[0][0][i]));
      avVarW[0][i]= varr(vW);       // Store particle weight variance
    }
  return meanr(avLL[0]);
}
```

Table 4: The particle filter in Ox.

for $\mathsf{P}(\theta|Y_T)$ at the posterior mode. Evaluating this approximation at the posterior mode itself gives $\mathsf{P}(\tilde{\theta}|Y_T) \approx (2\pi)^{-k/2}|\Sigma_\theta|^{-1/2}\exp(-\frac{1}{2}\times(\tilde{\theta}-\tilde{\theta})) = (2\pi)^{-k/2}|\Sigma_\theta|^{-1/2}$, with k the dimension of the parameter vector and Σ_θ the posterior covariance, see also Table 5. For small dimensions of the parameter vector and near-Gaussian posterior densities, this Laplace approximation of the marginal likelihood can be sufficiently precise (Kass, Tierney, and Kadane 1990; Bos 2002).

The Bayes factor B_{12} (Jeffreys 1961) is calculated using the marginal likelihood from two competing models $\mathcal{M}_1$ and $\mathcal{M}_2$, as in

$$B_{12} = \frac{M(Y|\,\mathcal{M}_1)}{M(Y|\,\mathcal{M}_2)}.$$

This odds ratio is the factor by which the data considers $\mathcal{M}_1$ more probable than the alternative $\mathcal{M}_2$. Kass and Raftery (1995) consider twice the difference in log-marginal likelihoods,

```
MargLiklLP(const vMode, const mS2, const dLnPdf, const mPrior)
{
  decl dDS2, dLnPost, iK, i;

  iK= sizerc(vMode);
  dDS2= determinant(mS2);

  dLnPost= dLnPdf;
  for (i= 0; i < iK; ++i)
    dLnPost+= lndensigamma1(vMode[i], mPrior[i][0], mPrior[i][1]);

  return (iK/2)*log(M_2PI)+0.5*log(dDS2)+dLnPost;
}
```

Table 5: Marginal loglikelihood using Laplace approximation in Ox.

	θ_0	r	a	$\overline{\theta}$	σ_θ
σ_ε	120	2.66	30,000	125	(50)
σ_ξ	30	2	5,000	63	(33)

Table 6: Starting values and prior density for parameters Nile data.

and describe that a difference of $2 < 2B_{12} < 6$ gives 'positive evidence' for model 1, and $2B_{12} > 10$ gives 'very strong evidence'. These marginal likelihoods and their differences will be used in Section 3 and later to decide whether there is evidence in the data for the inclusion of a stochastic volatility effect in the model.

2.5. Empirical results on samplers for Nile river flow

Both the Metropolis and the Gibbs sampler depend on the specification of the prior density for the parameters. The conjugate IG-1 density was chosen for the two standard deviations in the model. Table 6 provides the starting values θ_0, the parameters of the IG-1 prior density, and the corresponding prior mean $\overline{\theta}$ and standard deviation σ_θ. The priors are chosen such that their expectation is somewhat in the neighbourhood of the classical estimates, with standard deviations which allow for a considerable amount of uncertainty. For the Metropolis sampler, the random walk standard deviation is taken to be one tenth of the prior standard deviation.

With these values, the two samplers were run, both first for 10,000 iterations to allow the sampler to burn-in, then for 100,000 further iterations to collect the final sample. This was found to be a sufficiently large sample to obtain convergence. The posterior mean $\overline{\theta}$, standard deviation σ_θ, and inefficiency measure R_B of the parameters are displayed in Table 7, together with maximum likelihood estimates $\hat{\theta}, \sigma_\theta$ of the parameters in the first two columns. Figure 1 plots the posterior densities together with the prior densities and the autocorrelations of the sample.

	ML		MH			Gibbs		
	$\hat{\theta}$	σ_θ	$\overline{\theta}$	σ_θ	R_B	$\overline{\theta}$	σ_θ	R_B
σ_ε	122.876	(12.81)	118.799	(10.90)	[57.7]	118.694	(11.10)	[4.5]
σ_ξ	38.332	(16.72)	47.665	(11.31)	[90.5]	48.011	(11.65)	[12.9]
Loglikelihood KF	-632.546							
Loglikelihood PF	-632.669		-632.723			-632.639		
Acceptance rate			0.792					
Duration	0.53s		2.66s			4.75s		

Table 7: Estimation results of local level model on Nile data.

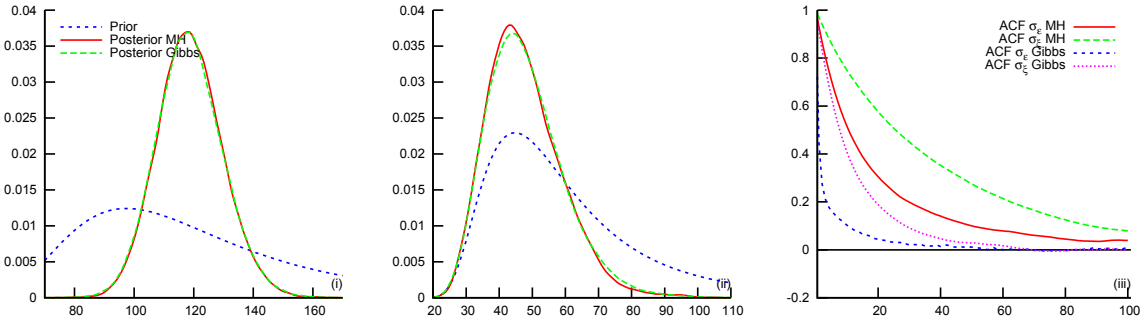

Figure 1: Prior and posterior densities for σ_ε (panel (i)) and σ_ξ (panel (ii)) for the Nile data, with autocorrelations in panel (iii), comparing MH and Gibbs results.

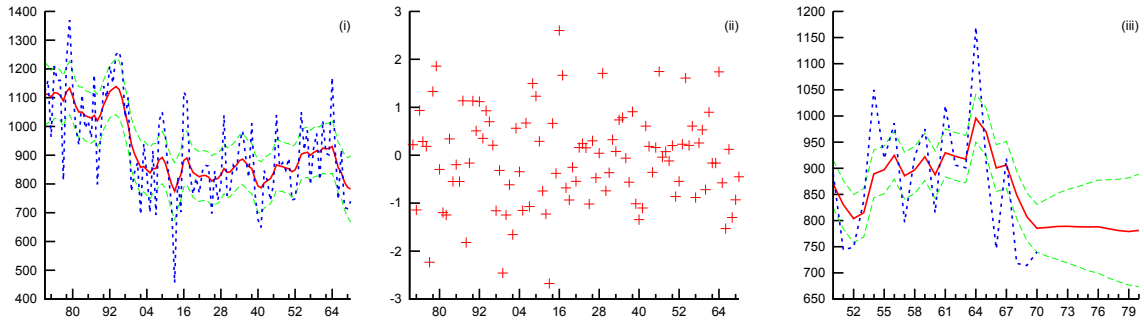

Figure 2: Nile data, with smoothed state and 90% confidence interval (panel (i)), standardized prediction errors (panel (ii)), and filtered state plus forecast with 50% confidence interval (panel (iii)).

The Nile data is found to be little informative on the precise location of the parameters, with large standard deviations for σ_ε and σ_ξ. Figure 1 shows that, especially for the state transition standard deviation σ_ξ, the prior influences the location of the posterior. The posterior densities of the MH and Gibbs samplers are similarly shaped, though the autocorrelation functions of the sample show how the Gibbs chain in this case delivers a sample with lower autocorrelation between successive draws. This lower autocorrelation also is apparent from the values for the inefficiency measures R_B in columns 5 and 7 of Table 7. These inefficiency measures display the relative variance of the posterior sample draws when adapting for the correlation between iterations, as compared to the variance without accounting for correlation. In these

calculations, a bandwidth B of 10% of the sample size is used. See Kim, Shephard, and Chib (1998) or Geweke (1992) for a further background on these measures.

Table 7 also reports the loglikelihood as it is calculated through the Kalman filter, and the value that results from the particle filter of Section 2.4, using 10,000 particles. The likelihood of the particle filter is virtually equal to the one computed using the Kalman filter, with the difference due to remaining sampling variation.

The fifth row displays the acceptance rate of the random walk Metropolis candidate drawings, which is 80%. The duration of the programs (on a 2.67Ghz Xeon X5550 processor) indicates that for such a small data set, running time is not a problem.

Figure 2 displays the Nile data together with the smoothed state estimate and the 90% confidence interval of the state in the first panel; this graph should be directly comparable to the smoothed estimates in other articles in the present special issue. The standardized prediction errors in the second panel stem from output of the particle filter: Using the filtered state, a further prediction step is taken, and the standardized prediction errors are extracted. The last panel displays the filtered state since 1950, extending the sample until 1980 to provide a forecast with a 50% confidence bound.

3. Introducing time varying volatility

3.1. Extending the model

The unobserved components model for the Nile data as described above is a linear Gaussian state space model. This simplifies the analysis considerably, as the Kalman filtering equations can be used directly. One could however wonder whether the volatility of the annual flow volume of the Nile for example is constant over time. As the flow will depend on matters like population, water use for irrigation, or the construction of dams in the river, one could expect changes in the process of the flow series.

Note that looking back at Figure 2, for example, where the data itself was plotted, does not give a clear indication of changes in volatility. Besides, the number of observations ($T = 100$) is relatively low for detecting variance changes. Hence this second case will serve as a preparation for the last application, where volatility in financial returns is studied.

To introduce a simple variant of the local level model, with a second unobserved state for the stochastic volatility, the observation equation (1) is adapted to

$$y_t = \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim \text{NID}(0, \sigma_\varepsilon^2 \exp(h_t)). \tag{1'}$$

A second state equation is added, describing the evolution of the (log) volatility h_t as

$$h_{t+1} = h_t + \nu_t \qquad \nu_t \sim \text{NID}(0, \sigma_\nu^2), \tag{6}$$

initializing $h_1 \equiv 0$. With this specification, the initial variance of the observation equation is set to σ_ε^2, whereas at later time periods the log-variance can change according to a random walk.

3.2. Implementing the extended sampler

Using the Gibbs approach with data augmentation, we have a parameter vector consisting of $\theta = (\sigma_\varepsilon, \sigma_\xi, \sigma_\nu)$ together with state variables μ and h. Conditional on h and σ_ν, the previous Gibbs sampler works well, as the unobserved components model is still conditionally linear and Gaussian. The algorithm however needs two additional steps, one for simulating $h|Y_T, \mu, \theta$ and one for $\sigma_\nu|Y_T, \mu, h, \sigma_\varepsilon, \sigma_\xi$. The adapted sampling scheme becomes:

1. Initialize, start with a drawing $\theta^{(0)}, \mu^{(0)}$ and $h^{(0)}$, and set $i = 1$;
2. Sample $\mu^{(i)} \sim \mathsf{P}(\mu|Y_T, h^{(i)}, \theta^{(i)})$ using the simulation smoother, taking the time varying variances into account;
3. For $t = 2, \ldots, T$, the single-site volatility sampler draws successively
$$h_t^{(i)} \sim \mathsf{P}(h_t|h_{t-1}^{(i)}, h_{t+1}^{(i-1)}, y_t, \mu_t^{(i)})$$
using an acceptance-rejection scheme (Kim *et al.* 1998). This is performed by sampling a candidate h_t^c using
$$h_t^c \sim \text{NID}\left(h_t^* + \frac{1}{2}v^2\left(\left(y_t - \mu^{(i)}\right)^2 \exp(-h_t^*) - 1\right), v^2\right),$$
$$h_t^* \equiv \frac{1}{2}\left(h_{t-1}^{(i)} + h_{t+1}^{(i-1)}\right), \qquad v^2 = \frac{1}{2}\sigma_\nu^2.$$
Such a candidate draw is accepted with probability
$$\alpha(h_t^c) = \frac{\exp\left(-\frac{1}{2}\left(y_t - \mu_t^{(i)}\right)^2 \exp(-h_t^c)\right)}{\exp\left(-\frac{1}{2}\left(y_t - \mu_t^{(i)}\right)^2 (\exp(h_t^*)(1 + h_t^*) - h_t^c \exp(-h_t^*)\right)}.$$
If the candidate h_t^c is accepted, set $h_t^{(i)} = h_t^c$, else continue drawing a new candidate h_t^c. Note that when $t = T$, there is no h_{t+1} available, and hence h_t^* and v^2 have to be adapted slightly;
4. Sample $\sigma_\varepsilon^{(i)} \sim \text{IG-1}\left(r = r_\varepsilon + \frac{T}{2}, a = a_\varepsilon + \frac{1}{2}\sum \frac{\left(y_t - \mu_t^{(i)}\right)^2}{\exp(h_t^{(i)})}\right)$;
5. Sample $\sigma_\xi^{(i)} \sim \text{IG-1}\left(r = r_\xi + \frac{T-1}{2}, a = a_\xi + \frac{1}{2}\sum_{i=2}^{T}\left(\mu_t^{(i)} - \mu_{t-1}^{(i)}\right)^2\right)$;
6. Sample $\sigma_\nu^{(i)} \sim \text{IG-1}\left(r = r_\nu + \frac{T-1}{2}, a = a_\nu + \frac{1}{2}\sum_{i=2}^{T}(h_t^{(i)} - h_{t-1}^{(i)})^2\right)$;
7. Increase i, and repeat from Step 2. until the sample is sufficiently large.

Implementing Step 3 is relatively straightforward. Table 8 gives an example implementation of this single-site volatility sampler for h_t. Notice however that the present algorithm now contains an outer loop to sample a large collection of parameters, and within an inner loop over t it samples h. This implies that the code within the single-site sampler is executed very

```
SampleHAR(const avH, const vE, const dSNu, const iMax)
{
  ...
  dV2= sqr(dSNu)/2;
  iC= 0;
  for (i= 1; i < iT; ++i)
    { // Set moments
      if (i == iT-1)
        { // last observation, single-sided
          dHx= avH[0][i-1];
          dV2= sqr(dSNu);
        }
      else
        dHx= 0.5*(avH[0][i-1] + avH[0][i+1]);
      dMu= dHx + 0.5*dV2*(sqr(vE[i]) *exp(-dHx) - 1);

      iC0= 0;
      do
        { // Repeat until acceptance
          dHC= dMu + sqrt(dV2) * rann(1, 1);
          dLnFx= -0.5*sqr(vE[i]) * exp(-dHC);
          dLnGx= -0.5*sqr(vE[i]) * (exp(-dHx)*(1+dHx)-dHC*exp(-dHx));
          dU= ranu(1, 1);
          ++iC0;              // Count acceptances + rejections
        }
      while ((dU >= exp(dLnFx - dLnGx)) && (iC0 < iMax));
      avH[0][i]= dHC;
      iC+= iC0;
    }
  return (iT-1) / iC;       // Acceptance rate
}
```

Table 8: The single-site volatility sampler for $h_t|h_{t-1}, h_{t+1}, e_t$ in Ox.

often. The routine in Table 8 would be a first candidate for moving into a lower-level language like C, in order to decrease the running time of the algorithm considerably. This however lies beyond the scope of the present article.

Table 8 literally follows the proposed algorithm for sampling $h_t|h_{t-1}, h_{t+1}, y_t, \theta$. New candidate values h_t^c should be drawn until acceptance occurs. In some cases (especially for an unlucky choice of starting values) the probability of acceptance can become very low, and it is better to force acceptance after a limited number of trials. Here, the sampling continues until at most a total of `iMax` candidates have been drawn. In well-specified situations, the acceptance rate tends to be above 95%, and hence the `do` loop does not tend to reach a forced end in normal situations.

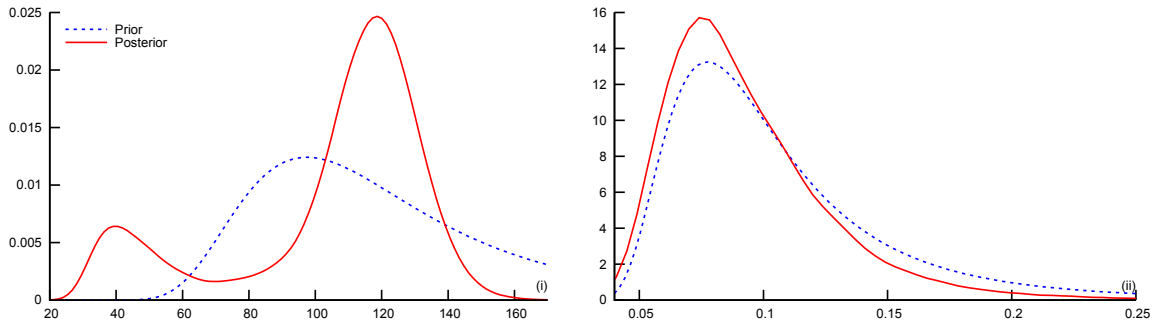

Figure 3: Prior and posterior density of σ_ε (panel (i)), and σ_ν (panel (ii)).

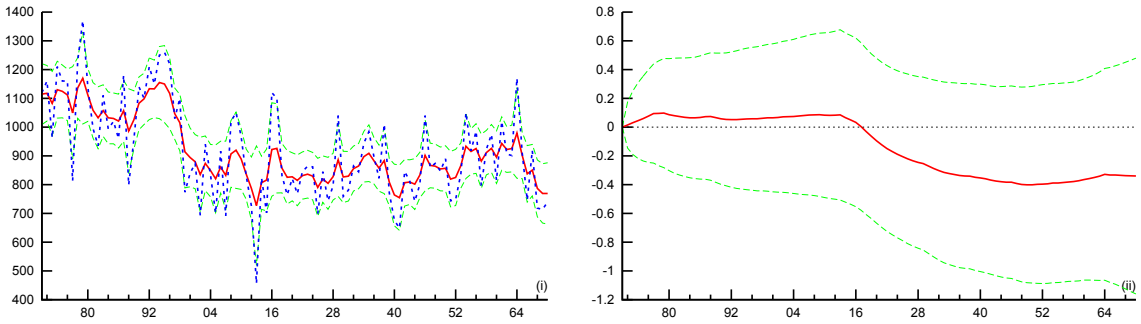

Figure 4: Nile data with smoothed level states and 90% confidence interval (panel (i)), and smoothed volatility with confidence interval (panel (ii)).

3.3. Empirical results on the volatility of Nile data

Some output of the sampler is provided in Figures 3–4. Again, a sample of 100,000 iterations was collected, after a burn-in of 10,000 iterations. The same priors as in Table 6 were used, with additionally $\pi(\sigma_\nu) = \text{IG-1}(r = 2, a = 0.015)$, implying that $E(\sigma_\nu) \approx 0.1$ with prior standard deviation $\sigma(\sigma_\nu) \approx 0.05$.

The posterior density of σ_ε indicates already that this extensive time varying volatility model might be too much to estimate from this small data set. The posterior density is bimodal, implying that apart from the solution with $\sigma_\varepsilon \approx 120$ an alternative parameter setting with $\sigma_\varepsilon \approx 40$ (and $\sigma_\xi \approx 150$, not displayed) could work: In that case, the state effectively follows the observations very closely, with the observation equation allowing extra variability for the outliers. In panel (ii) the standard deviation σ_ν of the stochastic volatility increments is depicted. As it was already hard to estimate a constant level of volatility, estimating volatility of volatility is too much using a data set of 100 observations, and the posterior closely follows the prior density.

Table 9 displays the posterior mode, the standard deviation and the inefficiency measure, both for the model without stochastic volatility of the previous section and for the model with time varying variance. The common parameters change little, though the uncertainty on the parameters increases strongly. The large inefficiency measures for the LL-SV model indicate that correlation in the chain is high, and effectively imply that the sampler did not converge; cf. the bimodal posterior distribution for σ_ε.

	LL			LL-SV		
	$\tilde{\theta}$	σ_θ	R_B	$\tilde{\theta}$	σ_θ	R_B
σ_ε	121.079	(11.10)	[4.5]	116.027	(29.88)	[1266.9]
σ_ξ	45.392	(11.65)	[12.9]	41.522	(39.53)	[1263.8]
σ_ν				0.073	(0.04)	[340.2]
Marginal loglikelihood		−634.47			−633.77	
Duration		4.25s			29.16s	

Table 9: Estimation results comparing local level model without/with stochastic volatility, on Nile data.

The standard deviation σ_ν of the volatility equation is small, which is consistent with a smooth volatility process. The row with the duration of the programs indicates that sampling the volatility process is relatively costly, as a separate sampling step for each $h_t|h_{t-1}, h_{t+1}, y_t, \theta$ has to be performed. The table includes a row reporting the marginal loglikelihood (see Section 2.4). In this case there is only a minor difference between the two marginal loglikelihoods, implying that the model with stochastic volatility does not fit the data much better than the model with fixed volatility.

Figure 4 displays the smoothed level of the Nile data, in panel (i), and the extracted volatility process, in panel (ii), both with 90% confidence intervals. Volatility appears constant until around 1914, moving to a slightly lower level afterwards. Uncertainty about the volatility is large, as was to be expected. It is unclear what exactly causes the drop in volatility. One possible explanation is the building and extension of the Aswan dam. In 1912 an improvement to the dam was made, which could have led to a better regulation of the river, leading to lower volatility in the flows. The change in volatility however is minor, consistent with the finding that the marginal likelihoods of the models with/without SV are virtually the same.

This section illustrates the relative ease with which the standard linear Gaussian unobserved components model is extended to the case with stochastic volatility. Conditional on the volatility, the sampler hardly changes; the only real change in the algorithm is the inclusion of a sampler for the volatilities, and for the parameter σ_ν in the volatility process.

Note that the present single-site volatility sampler is a relatively simple sampler. It may lead to extensive correlation in the sampling chain as each of the $T-1$ volatilities h_t is sampled successively, conditionally on past and future values h_{t-1}, h_{t+1}. Alternative approaches, like the mixture sampler of Kim *et al.* (1998), could lower correlation between the draws in the Markov chain.

4. Time varying volatility for S&P 500 returns

4.1. Introducing S&P 500 data

Figure 5 displays the S&P 500 index data (source: Yahoo finance) for the period 2000/1/3–2009/12/30. The first panel displays the level of the series, followed by the returns in panel (ii). From this second panel the strong differences in volatility are immediately apparent. The third panel plots the average return and the average standard deviation of the S&P 500 series, using a running window of one year. For the variance, the tranquil period between 2004–2008

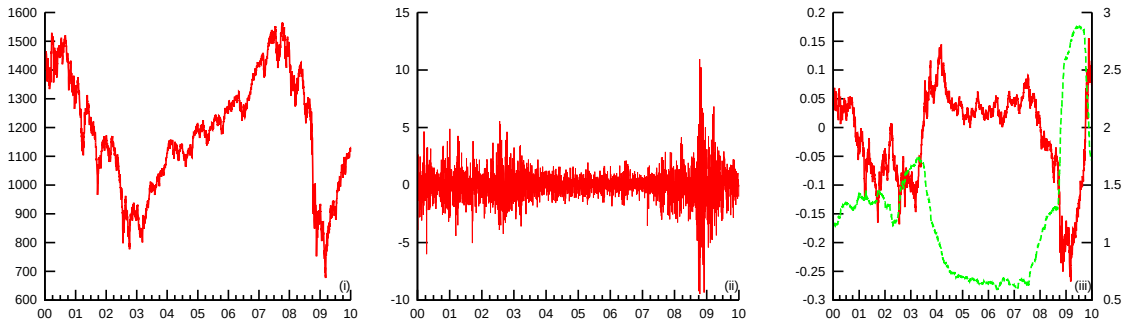

Figure 5: S&P 500 data over 2000–2009 in levels (panel (i)), returns (ii), and the yearly average return and standard deviation (iii).

	θ_0	r	a	$\overline{\theta}$	σ_θ
σ_ε	1	2	1.5	1.1	0.57
σ_ξ	0.001	2	0.00015	0.011	0.0057
σ_ν	0.1	2	0.015	0.11	0.057

Table 10: Starting values and prior density for parameters S&P500 data.

is clear, with a strong increase in variance over the last two years of the sample. The average yearly return depicts roughly the opposite movement: When volatility is low, average returns were higher, and vice versa especially in the period after 2008. The question is whether a local level with stochastic volatility model on the returns of the S&P 500 would be able to recover these two movements. Notice that in this case the volatility is the clear signal, as that is easily detected from the movements in panel (ii). The differing average returns are not apparent from the middle panel, and hence will be harder to estimate.

4.2. Volatility estimation on S&P 500 returns

The sampler from the previous section works without change for the present data set. The only change needed in the program is the specification of the prior and the starting values of the parameters. Table 10 displays these specifications for the S&P 500 data, cf. Table 6.

Results on the posterior sample are gathered in Table 11, again comparing the local level model with the model including stochastic volatility. In both cases, a Gibbs sampler of 10,000 burn-in iterations, with a final sample size of 100,000 parameter vectors is run, followed by a particle filter with 10,000 particles as before. Reported in the table are the posterior mode $\tilde{\theta}$, the posterior standard deviation σ_θ, and the inefficiency measure R_B at a bandwidth of 10% of the sample.

We first discuss the estimation results of the parameters. For the common parameters σ_ε and σ_ξ, inclusion of the SV sequence makes little difference in the estimated values. The chain delivers strongly correlated drawings from σ_ξ, σ_ν, as indicated by the high inefficiency values for these parameters. Estimation using the Gibbs sampler took 8.5 times as long when the stochastic volatility sequence was introduced: Sampling volatility is computationally hard.

Figure 6 displays the prior and posterior densities of these parameters, together with the

	LL			LL-SV		
	$\tilde{\theta}$	σ_θ	R_B	$\tilde{\theta}$	σ_θ	R_B
σ_ε	1.39994	(0.0197)	[0.5]	1.39755	(0.0198)	[0.0]
σ_ξ	0.00622	(0.0018)	[399.5]	0.00544	(0.0013)	[390.2]
σ_ν				0.10866	(0.0120)	[882.5]
Marginal loglikelihood		-4424.44			-3783.28	
Duration (Gibbs/PF)		1m29s/13s			12m41s/15s	

Table 11: Estimation results for S&P 500 data.

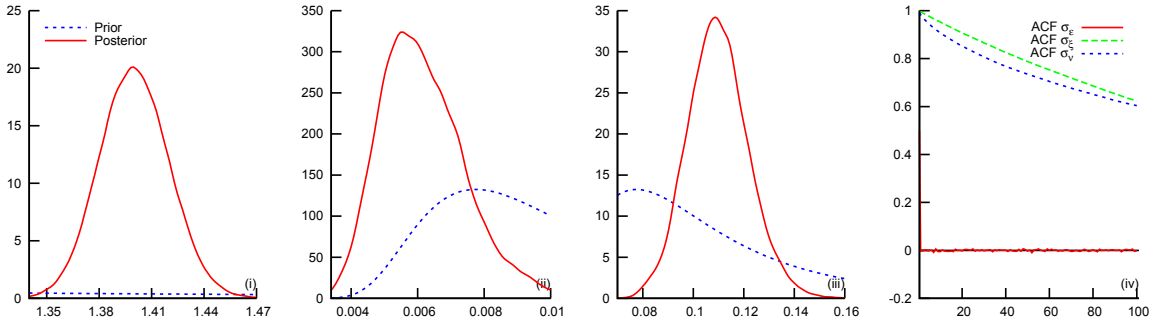

Figure 6: Prior and posterior density for σ_ε (panel (i)), σ_ξ (ii), σ_ν (iii) and the autocorrelations of the parameters (iv), for S&P 500 data.

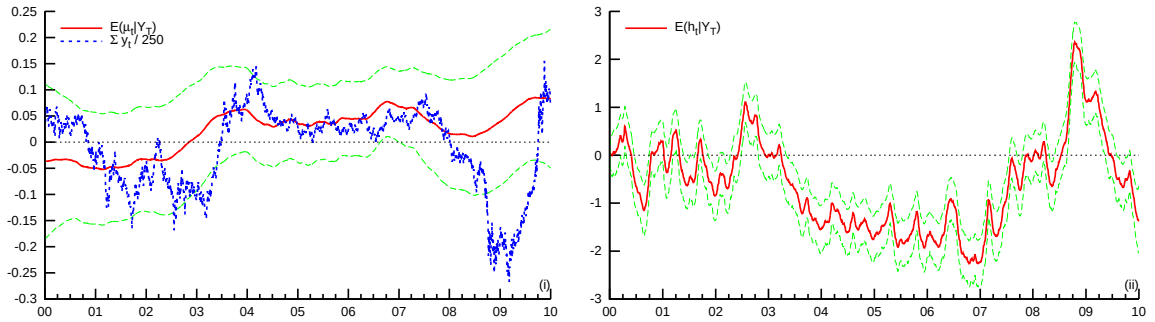

Figure 7: S&P 500 smoothed states, for the level (panel (i)), with 90% confidence bounds and with the average yearly return superimposed, and for the volatility (panel (ii)).

autocorrelation functions, effectively giving the same information as the table. Here it is even clearer that the transition standard deviation, in panel (ii), is hard to estimate, with the posterior pushing downwards in comparison with the prior. The prior is informative in this situation, as it forces the posterior away from zero.

The standard deviation σ_ξ determines the smoothness of the state variable μ_t in the model. Panel (ii) in Figure 7 displays the smoothed state estimate, which indeeds indicates what periods tended to display positive or negative returns. The present estimate however is far smoother than the running yearly average return. The 90% confidence bounds include the value of zero over the full sample, indicating that at no period in time the expected return was clearly positive or negative.

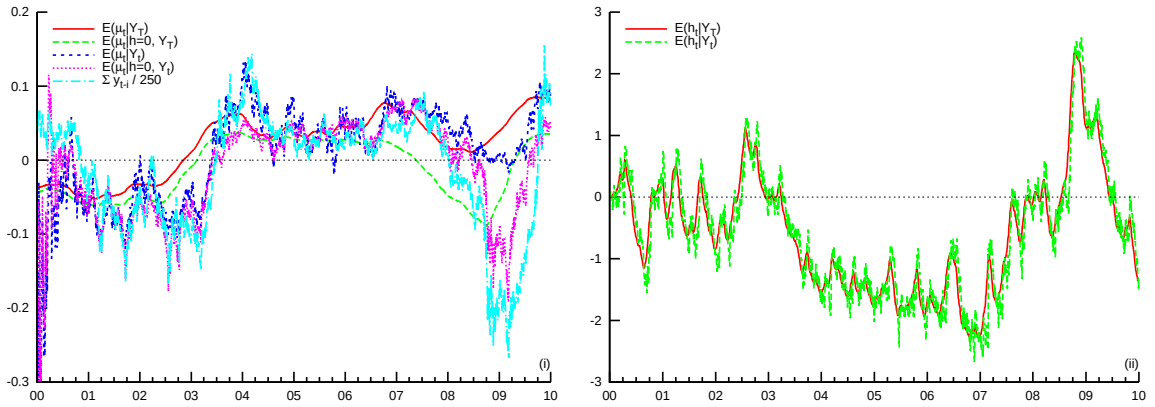

Figure 8: Comparing smoothed and filtered state estimates, for the level (panel (i)) and volatility (panel (ii)), according to the LL ($h \equiv 0$) and LL-SV models.

A different prior specification for σ_ξ could force the level to be more or less smooth, by putting more prior weight on smaller or larger values of σ_ξ than the present choice of prior. Depending on the purpose of extracting the expected returns, the flexibility of the Bayesian approach can be an advantage in this respect, extracting a smoother or more volatile signal.

For the S&P 500 returns, the volatility is clearly changing over time. The period 2004–2008 is found to display low variance, although the uncertainty slowly starts to rise in the second half of 2007. According to these estimates, the peak in uncertainty was reached in the beginning of 2009, with volatility retreating towards levels comparable to the years 2004-2007 at the end of the sample.

In Section 2.4 a particle filter for the linear state space model was introduced. Extending it to allow for time varying volatility is straightforward: Sample new volatilities $h_t^{(i)} \sim \text{NID}(h_{t-1}^{(i)}, \sigma_\nu^2)$ in each step, and adapt for the volatility in the calculated weights. Figure 8 displays the results for the states using 10,000 particles, evaluated at the posterior mean estimates of the parameter as in Table 11.

Panel (i) compares the smoothed level from the Gibbs sampler, $E(\mu_t|Y_T)$, to the smoothed level from the Gibbs sampler for the LL model, $E(\mu_t|h = 0, Y_T)$, as well as to the filtered levels $E(\mu_t|Y_t)$ and $E(\mu_t|h = 0, Y_t)$. Additionally, the average yearly return, $\sum_{i=0}^{249} y_{t-i}/250$ is plotted. For the first part of the sample, both the particle filtered states, $E(\mu_t|Y_t)$ and $E(\mu_t|h = 0, Y_t)$, and the data-based average yearly return are fairly similar in structure. It is only at the advent of the financial crisis that the estimates start to differ. As the volatility increases, the signal-to-noise ratio drops strongly, and hence after 2008 the particle filter for the LL-SV model does not adapt the state estimate much. The estimate $E(\mu_t|h = 0, Y_T)$ is adapted much more when the constant variance LL model is used.

Panel (ii) of Figure 8 compares the smoothed and filtered volatility estimates. Both estimates agree closely, with the filtered estimate naturally being less smooth. It is however rather remarkable to see how little difference there is between the volatility extracted conditional on the full data set, or conditioning on only past data. The variance σ_ν^2 of the volatility series is such that h_t is estimated mostly on the basis of present information, and both past and future information are less important. For the level, in the first panel, the difference between the smoothed and filtered estimates is far larger. As the data itself contains little information

on the expected returns, information over a longer horizon is combined. If future information is available, as in the smoothed estimate of the state, then this information can alter the estimate of the state considerably.

The particle filter could also be used to construct a likelihood estimate, which in turn could serve as input for the calculation of the marginal likelihood (see (5)). Table 11 reports the values for the marginal loglikelihoods. The LL-SV model has a marginal loglikelihood which is 641 points higher than that of the LL model. According to the scheme of Kass and Raftery (1995), this log-difference of 641 is an indication that stochastic volatility is an indispensable part of the model for S&P 500 returns.

5. Concluding remarks

The purpose of this article was to present methods, algorithms and implementations for analysing the Nile river flow data using Ox in combination with Bayesian Markov chain Monte Carlo techniques. Section 2 concludes that the Bayesian analysis is able to mimic closely the results from a classical analysis, so one could think that the Bayesian approach here is of little added value.

In Section 3 the extension towards a model with time varying variance is made. Within the Gibbs sampler, such an extension is easily implemented. From the results, some indication is found that volatility in the river flow dropped after 1916. Uncertainty about the volatility is large, as the number of observations is very limited. The marginal likelihood indicated that indeed the data does not favor a time varying volatility.

The posterior of the parameters in this case indicated multimodality of the density of σ_ε and σ_ξ. The Bayesian approach gives a clear signal that different parameter settings could describe the data well. In a classical approach, such multimodality is often hard to recognize, and it is not always clear whether reported maximum likelihood estimates are from the global or possibly from a local maximum.

The local level plus volatility model is applied in Section 4 on returns of the S&P500 data. The existing programs could be used with little change to extract a heavily time varying volatility for these financial index returns. For this series, extracting the expected return as measured by the local level of the model is harder, as this data is more informative on the volatility than on the level. The marginal likelihood in this case overwhelmingly favors the local level model with stochastic volatility, instead of the standard linear Gaussian unobserved components model with fixed variances.

In all these cases, an implementation in Ox was described in detail, and with short programs these extensive models could be estimated without further problems. This serves to show the flexibility of both the language and of the Bayesian MCMC approach applied here.

References

Bauwens L, Lubrano M, Richard JF (1999). *Bayesian Inference in Dynamic Econometric Models*. Advanced Texts in Econometrics. Oxford University Press, Oxford.

Bos CS (2002). "A Comparison of Marginal Likelihood Computation Methods." In W Härdle,

B Rönz (eds.), *COMPSTAT 2002 – Proceedings in Computational Statistics*, pp. 111–117. Physica-Verlag, Heidelberg.

Bos CS (2003). ***GnuDraw***. URL http://www.tinbergen.nl/~cbos/gnudraw.html.

Bos CS, Shephard N (2006). "Inference for Adaptive Time Series Models: Stochastic Volatility and Conditionally Gaussian State Space Form." *Econometric Reviews*, **25**(2–3), 219–244.

Cappé O, Godsill SJ, Moulines E (2007). "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo." *Proceedings of the IEEE*, **95**(5), 899–924.

Chib S, Greenberg E (1995). "Understanding the Metropolis-Hastings Algorithm." *The American Statistician*, **49**(4), 327–335.

Commandeur JJF, Koopman SJ, Ooms M (2011). "Statistical Software for State Space Methods." *Journal of Statistical Software*, **41**(1), 1–18. URL http://www.jstatsoft.org/v41/i01/.

Doornik JA (2009). *Ox 6: An Object-Oriented Matrix Programming Language*. Timberlake Consultants Ltd, London.

Durbin J, Koopman SJ (2001). *Time Series Analysis by State Space Methods*. Oxford University Press, Oxford.

Durbin J, Koopman SJ (2002). "A Simple and Efficient Simulation Smoother for State Space Time Series Analysis." *Biometrika*, **89**(3), 603–615.

Geweke J (1992). "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments." In JM Bernardo, JO Berger, AP Dawid, AFM Smith (eds.), *Bayesian Statistics 4: Proceedings of the Fourth Valencia International Meeting*, pp. 169–193. Oxford: Clarendon Press.

Harvey AC (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, Cambridge.

Jeffreys H (1961). *Theory of Probability*. 3rd edition. Oxford University Press, Oxford.

Kass RE, Raftery AE (1995). "Bayes Factors." *Journal of the American Statistical Association*, **90**(430), 773–795.

Kass RE, Tierney L, Kadane JB (1990). "The Validity of Posterior Expansions Based on Laplace's Method." In S Geisser (ed.), *Bayesian and Likelihood Methods in Statistics and Econometrics: Essays in Honor of George A. Barnard*, pp. 473–488. North Holland, Amsterdam.

Kim S, Shephard N, Chib S (1998). "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models." *Review of Economic Studies*, **65**(3), 361–393.

Koopman SJ, Shephard N, Doornik JA (1999). "Statistical Algorithms for Models in State Space Using **SsfPack** 2.2." *Econometrics Journal*, **2**(1), 107–160.

Koopman SJ, Shephard N, Doornik JA (2008). ***SsfPack** 3.0: Statistical Algorithms for Models in State Space Form*. Timberlake Consultants Ltd, London.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953). "Equations of State Calculations by Fast Computing Machines." *Journal of Chemical Physics*, **21**, 1087–1091.

Pitt MK, Shephard N (1999). "Filtering via Simulation: Auxiliary Particle Filters." *Journal of the American Statistical Association*, **94**(446), 590–599.

Smith AFM, Gelfand AE (1992). "Bayesian Statistics without Tears: A Sampling-Resampling Perspective." *The American Statistician*, **46**(2), 84–88.

A. The simulation smoother

In Durbin and Koopman (2002) a simple and efficient simulator for the states of a model in state space is proposed. It can be implemented using a few lines of code in Ox as in Table 12. Such code follows Algorithm 1 of Durbin and Koopman (2002) closely:

1. In the first step, a set of random disturbances is generated, which afterwards are run through the recursion of (1)–(2) using the function `SsfRecursion()` from `SsfPack`. Note that here the initial standard deviation should be provided in the last function argument, instead of the variance in `mSigma`. This results in pseudo-data y^+ and α^+ in the matrix `mR`.

2. Secondly, $y^* = y - y^+$ is calculated, the difference between the true and the pseudo-data. This pseudo-data is smoothed, using `SsfMomentEst()`, resulting in $\hat{\alpha}$.

3. The simulated states are now the difference between the pseudo-states α^+ and the smoothed states $\hat{\alpha}$. This difference is returned from the function.

Here, the simulation smoother is implemented for the most simple state space model. With minor changes similar code would work as well for models with e.g., time varying variances, with intercepts, or other extensions to the basic local level model.

B. List of programs and routines

The algorithms described in this article have been implemented in Ox (Doornik 2009), making use of a few routines in **SsfPack 2.2** (Koopman *et al.* 1999). For graphical output, the

```
SimSmoDK(const vY, const mPhi, const mOmega, const mSigma)
{
  decl vAlpha, mU, iT, mP, mR, mYStar, mD, ir;

  iT= sizerc(vY);                          // Generate new pseudo-data
  mU= rann(2, 1)~choleski(mOmega)*rann(2, iT);
  // Don't use diffuse initialization for recursion
  mP= mSigma[0][0] == -1 ? 1 : sqrt(mSigma[0][0]);
  mR= SsfRecursion(mU, mPhi, mOmega, mP|mSigma[1][]);

  mYStar= vY - mR[1][1:];              // Smooth the difference with y
  SsfMomentEst(ST_SMO, &mD, mYStar, mPhi, mOmega, mSigma[0][]|0);

  // The simulated state is the sum of pseudo-state and smoothed state
  return mR[0][:iT-1] + mD[0][:iT-1];
}
```

Table 12: The simulation smoother in Ox.

Section	File	Data	SV	Method
2.2	`estnilemh.ox`	Nile	-	Metropolis-Hastings
2.3	`estnilegb.ox`	Nile	-	Gibbs with data augmentation
2.4	`estnilepf.ox`	Nile	-	Particle filter
2.5	`estnileml.ox`	Nile	-	Maximum likelihood
3	`estnilegbsv.ox`	Nile	+	Gibbs with data augmentation
3	`estnilepfsv.ox`	Nile	+	Particle filter
4	`estspgb.ox`	S&P500	-	Gibbs with data augmentation
4	`estsppf.ox`	S&P500	-	Particle filter
4	`estspgbsv.ox`	S&P500	+	Gibbs with data augmentation
4	`estsppfsv.ox`	S&P500	+	Particle filter

Table 13: List of programs.

Table	Section	Routine	Purpose
1	2.2	`EstBayesMH`	The random walk Metropolis scheme
2	2.2	`LnPosterior`	Logarithm of the posterior density
3	2.3	`SimSigmaIG1`	Simulating from the IG-1 density
4	2.4	`PartFiltLL`	The particle filter
5	2.4	`MargLiklLP`	Marginal likelihood using Laplace
8	3.2	`SampleHAR`	The single-site sampler for $h_t \vert h_{t-1}, h_{t+1}, y_t, \theta$
12	A	`SimSmoDK`	The simulation smoother

Table 14: List of routines.

professional version of Ox is needed; alternatively, the **GnuDraw** package of Bos (2003) can be used.

The code for these algorithms is available at the website of the journal, together with the necessary data. The code is split into several self-contained programs, as listed in Table 13. The routines that have been discussed in more detail, including their Ox code, are listed in Table 14.

Affiliation:

Charles S. Bos
Tinbergen Institute Amsterdam
Gustav Mahlerplein 117
NL-1082 MS Amsterdam, The Netherlands
and

Department of Econometrics FEWEB
VU University Amsterdam
De Boelelaan 1105
NL-1081 HV Amsterdam, The Netherlands
E-mail: cbos@feweb.vu.nl
URL: http://www.tinbergen.nl/~cbos/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 41, Issue 13 *Submitted:* 2010-01-29
May 2011 *Accepted:* 2010-11-29